```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define L 20

struct node
{
    char word[L+1];
    struct node *left, *right;
};

typedef struct node Node;

Node *newNode(char *word)
{
    Node *tmp = (Node*) malloc(sizeof(Node));
    strcpy(tmp->word, word);
    tmp->left = NULL;
    tmp->right = NULL;
    return tmp;
}

Node **searchWord(Node **root, char *word)
{
    while (*root!=NULL)
    {
        if (strcmp(word, (*root)->word)<0)
            root = &(*root)->left;
        else if (strcmp(word, (*root)->word)>0)
            root = &(*root)->right;
        else
            break;
    }
    return root;
}

int insertWord(Node **root, char *word)
{
    root = searchWord(root, word);
    if (*root)
        return 0;
    *root = newNode(word);
    return 1;

}

int deleteWord(Node **root, char *word)
{
    void deleteNode(Node **root)
    {
        if ((*root)->left && (*root)->right)
        {
            Node **tmp = &(*root)->left;
            while ((*tmp)->right)
                tmp = &(*tmp)->right;
            Node *newRoot = *tmp;
            *tmp = newRoot->left;
```

```
            newRoot->left = (*root)->left;
            newRoot->right = (*root)->right;
            free(*root);
            *root = newRoot;
        }
        else if ((*root)->left && !(*root)->right)
        {
            Node *tmp = *root;
            *root = (*root)->left;
            free(tmp);
        }
        else if (!(*root)->left && (*root)->right)
        {
            Node *tmp = *root;
            *root = (*root)->right;
            free(tmp);
        }
        else
        {
            free(*root);
            *root = NULL;
        }
    }

    root = searchWord(root, word);
    if (!*root)
        return 0;
    deleteNode(root);
    return 1;
}

void inorder(Node *root)
{
    void _inorder(Node *root, int *p)
    {
        if (!root)
            return;
        _inorder(root->left, p);
        printf("%s%s", *p?" ":"", root->word);
        *p = 1;
        _inorder(root->right, p);
    }

    int p = 0;
    _inorder(root, &p);
    printf("\n");
}

void preorder(Node *root)
{
    void _preorder(Node *root, int *p)
    {
        if (!root)
            return;
        printf("%s%s", *p?" ":"", root->word);
        *p = 1;
        _preorder(root->left, p);
```

```
        _preorder(root->right, p);
    }

    int p = 0;
    _preorder(root, &p);
    printf("\n");
}

void postorder(Node *root)
{
    void _postorder(Node *root, int *p)
    {
        if (!root)
            return;
        _postorder(root->left, p);
        _postorder(root->right, p);
        printf("%s%s", *p?" ":"", root->word);
        *p = 1;
    }

    int p = 0;
    _postorder(root, &p);
    printf("\n");
}

int checkWord(char *word)
{
    while (*word)
    {
        if (*word<'a' || *word>'z')
            return 0;
        ++word;
    }
    return 1;
}

int main()
{
    int i, n;
    char cmd[L+1];
    Node *root = NULL;
    scanf("%d", &n);
    for (i=0; i<n; ++i)
    {
        scanf("%s", cmd);
        checkWord(cmd);
        insertWord(&root, cmd);
    }
    while (scanf("%s", cmd)==1)
    {
        if (strcmp(cmd, "INSERT")==0)
        {
            scanf("%s", cmd);
            checkWord(cmd);
            if (!insertWord(&root, cmd))
                printf("error\n");
        }
```

```
        else if (strcmp(cmd, "DELETE")==0)
        {
            scanf("%s", cmd);
            checkWord(cmd);
            if (!deleteWord(&root, cmd))
                printf("error\n");
        }
        else if (strcmp(cmd, "INORDER")==0)
            inorder(root);
        else if (strcmp(cmd, "PREORDER")==0)
            preorder(root);
        else if (strcmp(cmd, "POSTORDER")==0)
            postorder(root);
    }
    return 0;
}
```